1. Series and DataFrames

PanDas stands for **"Panel Data System"** and it's an open-source data analysis and manipulation tool.
It's built on top of **Numpy** library.

Two primary data structures used in Pandas,

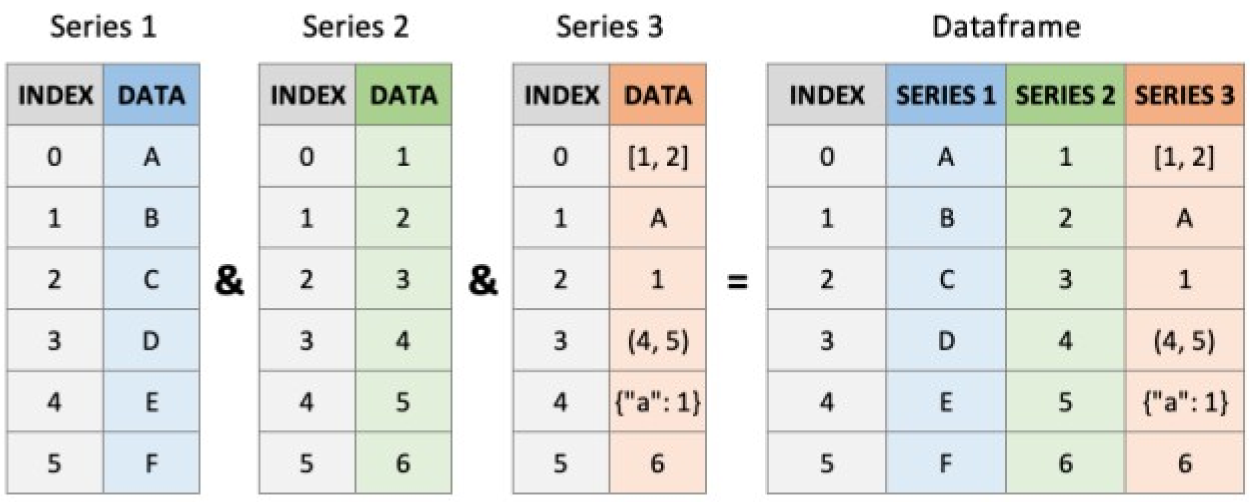
1. Series
2. DataFrames

Series

- A one-dimensional array like object(Imagine a vertical array).
- Able to hold any data type.
- Similar to a column in a Database table.
- Each element in a series is associated with an index(Used to uniquely identify elements). This index can be either a string or a number.

DataFrames

- Similar to a table in a relational database.
- A tabular data structure comprised of rows and columns.
- DataFrame can think of as a **group of Series** objects that share an index.



Series 1

INDEX	DATA
0	A
1	B
2	C
3	D
4	E
5	F

&

Series 2

INDEX	DATA
0	1
1	2
2	3
3	4
4	5
5	6

&

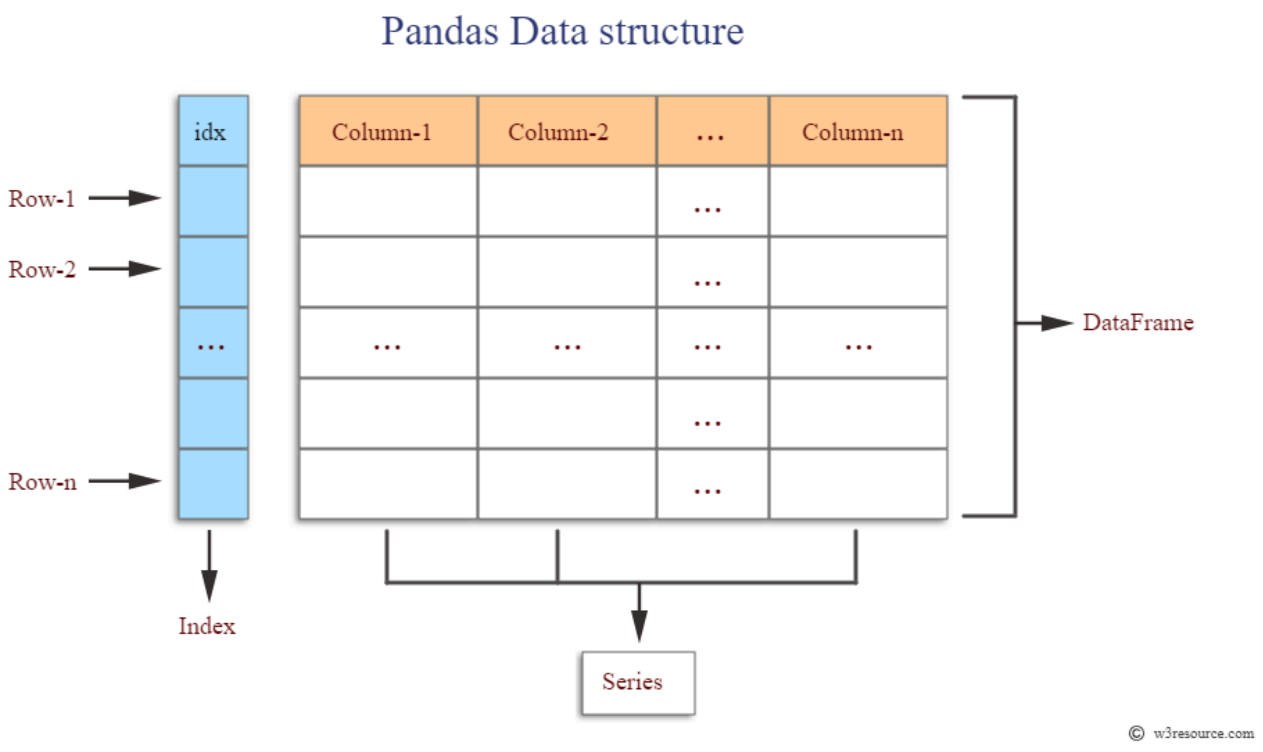
Series 3

INDEX	DATA
0	[1, 2]
1	A
2	1
3	(4, 5)
4	{"a": 1}
5	6

=

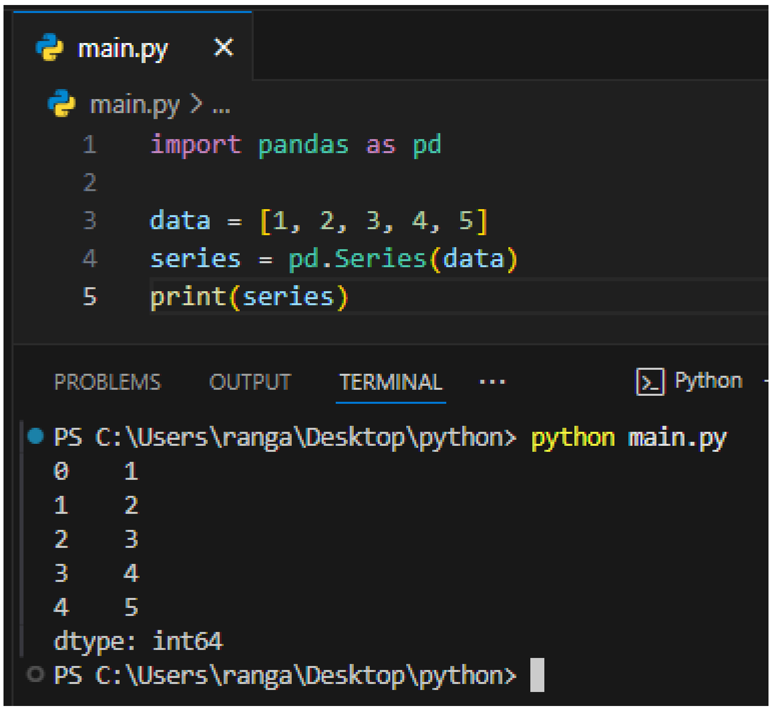
Dataframe

INDEX	SERIES 1	SERIES 2	SERIES 3
0	A	1	[1, 2]
1	B	2	A
2	C	3	1
3	D	4	(4, 5)
4	E	5	{"a": 1}
5	F	6	6



Creating Series and DataFrames :-

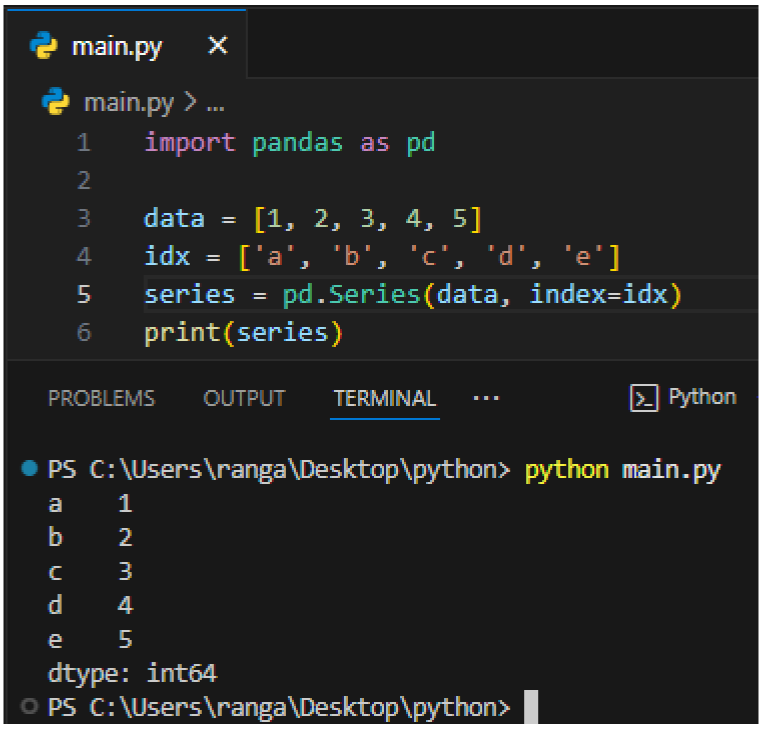
1. Create a Series from a list



```python
import pandas as pd

data = [1, 2, 3, 4, 5]
series = pd.Series(data)
print(series)
```

```
PS C:\Users\ranga\Desktop\python> python main.py
0    1
1    2
2    3
3    4
4    5
dtype: int64
PS C:\Users\ranga\Desktop\python>
```

2. Creating a Series with custom index labels

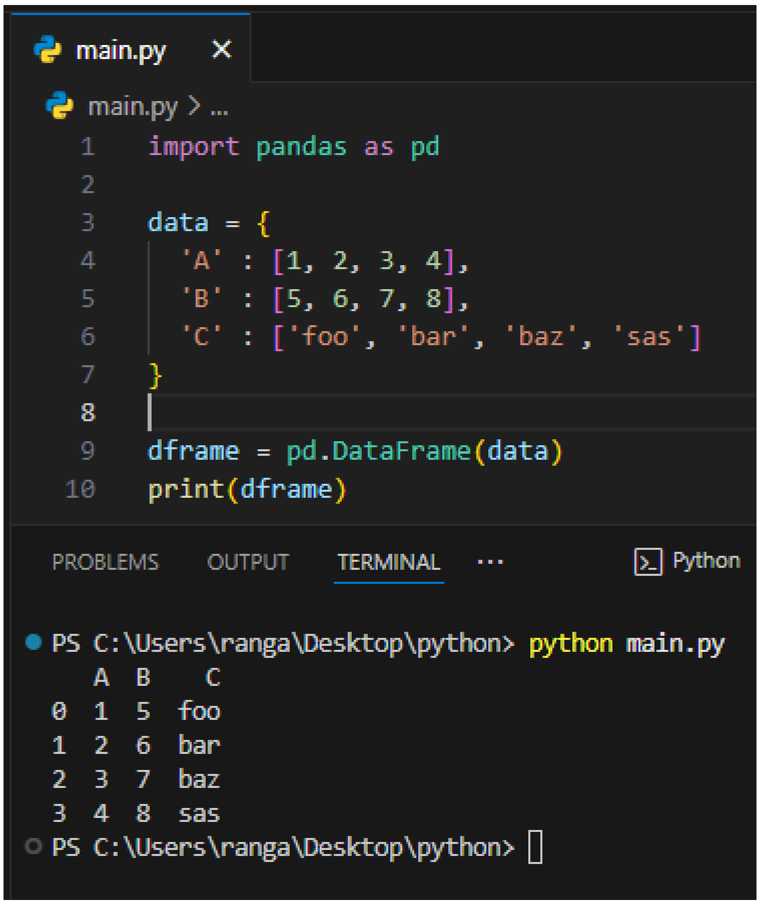


```python
import pandas as pd

data = [1, 2, 3, 4, 5]
idx = ['a', 'b', 'c', 'd', 'e']
series = pd.Series(data, index=idx)
print(series)
```

```
PS C:\Users\ranga\Desktop\python> python main.py
a    1
b    2
c    3
d    4
e    5
dtype: int64
PS C:\Users\ranga\Desktop\python>
```

3. Creating a DataFrame from a dictionary of lists

Python dictionary is a data structure that consist of key-value pairs.
Key can be used to identify each value and values can be any data type.



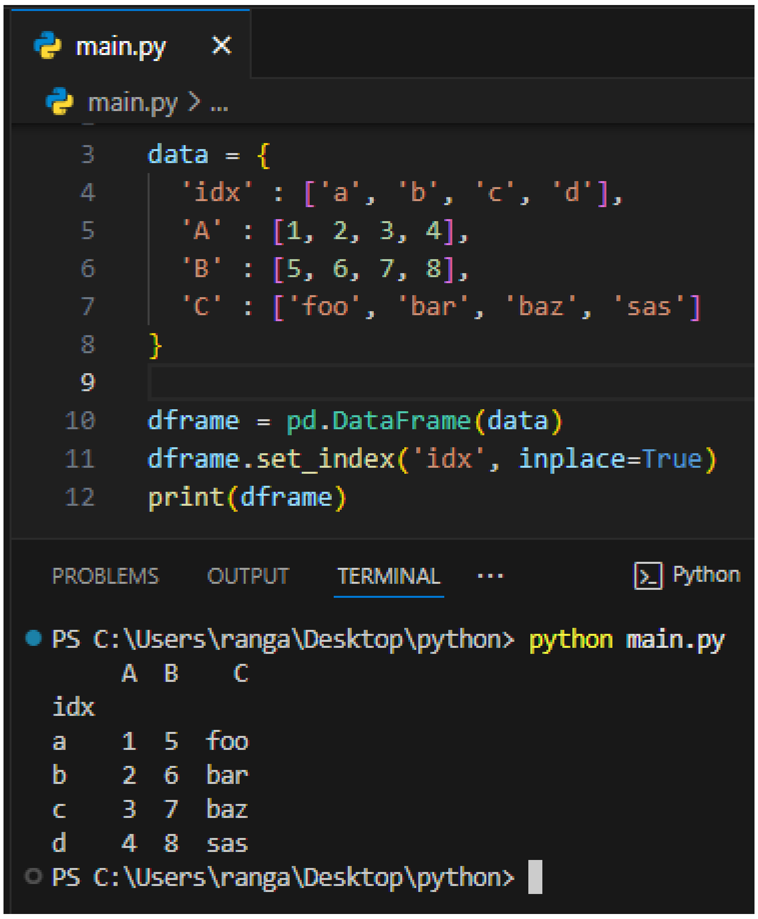
```python
import pandas as pd

data = {
    'A' : [1, 2, 3, 4],
    'B' : [5, 6, 7, 8],
    'C' : ['foo', 'bar', 'baz', 'sas']
}

dframe = pd.DataFrame(data)
print(dframe)
```

```
PS C:\Users\ranga\Desktop\python> python main.py
   A  B    C
0  1  5  foo
1  2  6  bar
2  3  7  baz
3  4  8  sas
PS C:\Users\ranga\Desktop\python>
```

Here,

- All **A, B, C** key-value pairs were used as Series.
- Index was auto-generated. `(0, 1, 2, 3)`
- Keys(`A,B,C`) considered as column names.
  We can tell panda to use one a key-value pair as the index of the DataFrame instead of

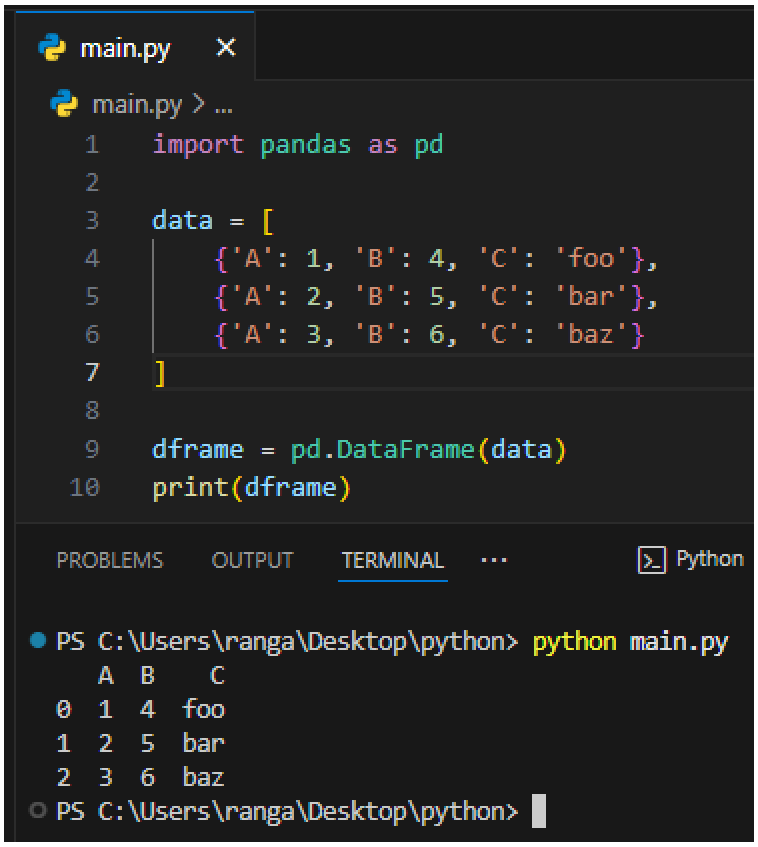
letting it generate automatically.



```python
data = {
    'idx' : ['a', 'b', 'c', 'd'],
    'A' : [1, 2, 3, 4],
    'B' : [5, 6, 7, 8],
    'C' : ['foo', 'bar', 'baz', 'sas']
}

dframe = pd.DataFrame(data)
dframe.set_index('idx', inplace=True)
print(dframe)
```

```
PS C:\Users\ranga\Desktop\python> python main.py
     A  B    C
idx
a    1  5  foo
b    2  6  bar
c    3  7  baz
d    4  8  sas
PS C:\Users\ranga\Desktop\python>
```

4. Creating a DataFrame from a list of dictionaries

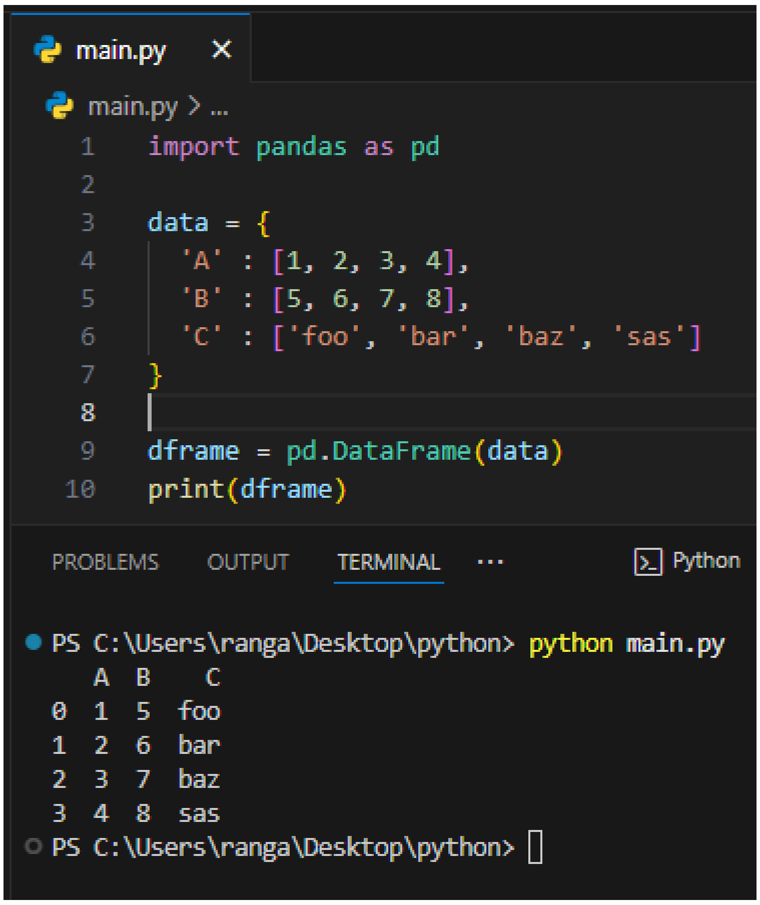


```python
import pandas as pd

data = [
    {'A': 1, 'B': 4, 'C': 'foo'},
    {'A': 2, 'B': 5, 'C': 'bar'},
    {'A': 3, 'B': 6, 'C': 'baz'}
]

dframe = pd.DataFrame(data)
print(dframe)
```

```
PS C:\Users\ranga\Desktop\python> python main.py
   A  B    C
0  1  4  foo
1  2  5  bar
2  3  6  baz
PS C:\Users\ranga\Desktop\python>
```

Additional :-

1. **set_index** function
   `set_index` function is used to set one or more columns of a DataFrame as the index.



main.py ×

main.py > ...

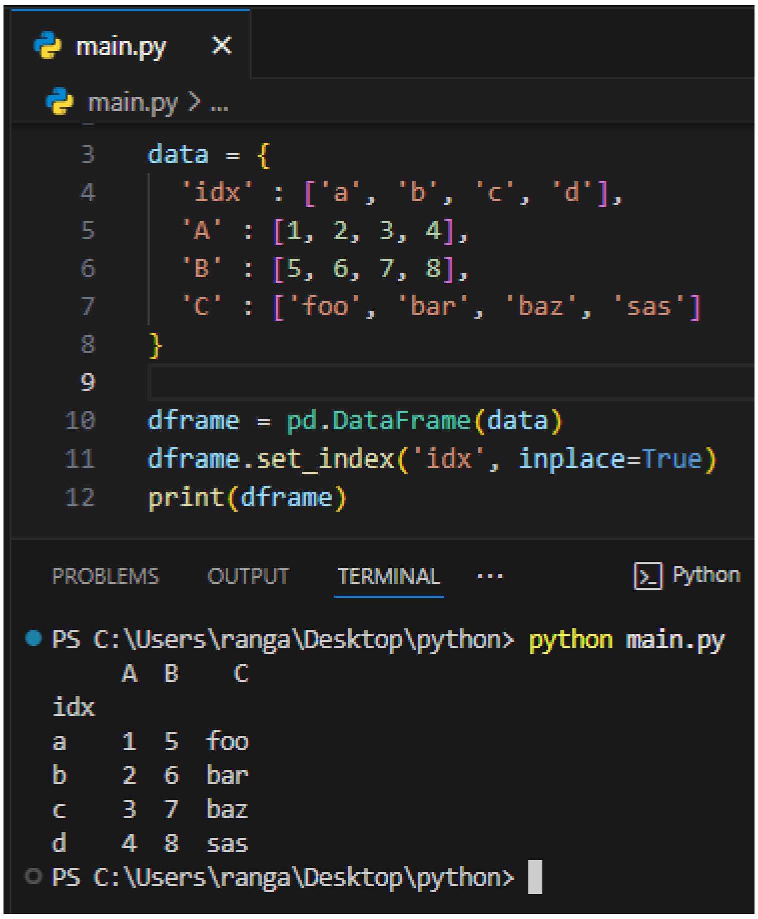
```python
import pandas as pd

data = {
    'A' : [1, 2, 3, 4],
    'B' : [5, 6, 7, 8],
    'C' : ['foo', 'bar', 'baz', 'sas']
}

dframe = pd.DataFrame(data)
print(dframe)
```

```
PS C:\Users\ranga\Desktop\python> python main.py
   A  B    C
0  1  5  foo
1  2  6  bar
2  3  7  baz
3  4  8  sas
PS C:\Users\ranga\Desktop\python> 
```

Here,

- All **A, B, C** key-value pairs were used as Series.
- Index was auto-generated. `(0, 1, 2, 3)`
- Keys(`A,B,C`) considered as column names.
  We can tell panda to use one a key-value pair as the index of the DataFrame instead of

letting it generate automatically.

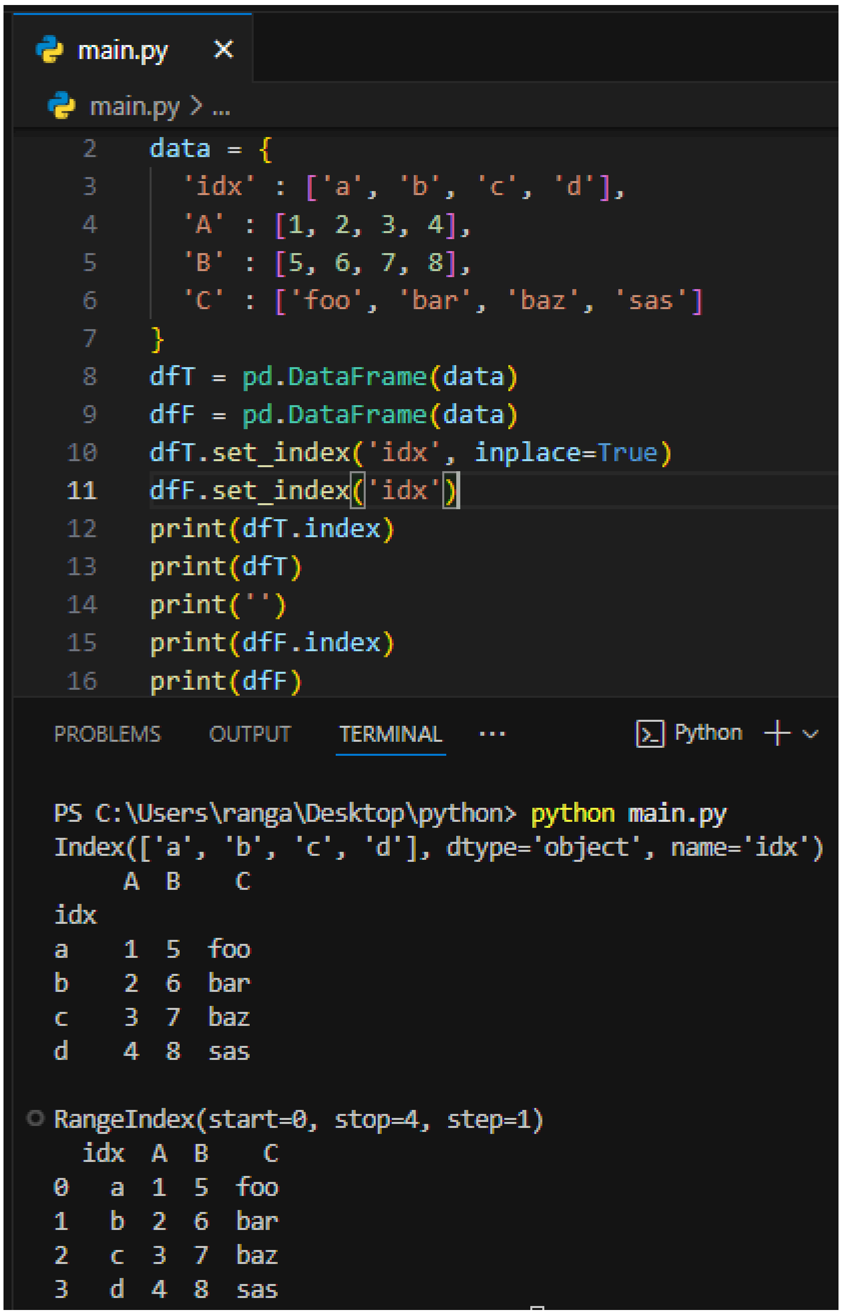


```python
data = {
    'idx' : ['a', 'b', 'c', 'd'],
    'A' : [1, 2, 3, 4],
    'B' : [5, 6, 7, 8],
    'C' : ['foo', 'bar', 'baz', 'sas']
}

dframe = pd.DataFrame(data)
dframe.set_index('idx', inplace=True)
print(dframe)
```

```
PS C:\Users\ranga\Desktop\python> python main.py
     A  B    C
idx
a    1  5  foo
b    2  6  bar
c    3  7  baz
d    4  8  sas
PS C:\Users\ranga\Desktop\python>
```

2. `inplace` parameter
   The `inplace` parameter is a Boolean flag that determines whether the operation is performed on the original DataFrame and then return the original DataFrame or the

modified DataFrame with changes.



```python
data = {
    'idx' : ['a', 'b', 'c', 'd'],
    'A' : [1, 2, 3, 4],
    'B' : [5, 6, 7, 8],
    'C' : ['foo', 'bar', 'baz', 'sas']
}
dfT = pd.DataFrame(data)
dfF = pd.DataFrame(data)
dfT.set_index('idx', inplace=True)
dfF.set_index('idx')
print(dfT.index)
print(dfT)
print('')
print(dfF.index)
print(dfF)
```

```
PS C:\Users\ranga\Desktop\python> python main.py
Index(['a', 'b', 'c', 'd'], dtype='object', name='idx')
     A  B    C
idx
a    1  5  foo
b    2  6  bar
c    3  7  baz
d    4  8  sas

RangeIndex(start=0, stop=4, step=1)
  idx  A  B    C
0   a  1  5  foo
1   b  2  6  bar
2   c  3  7  baz
3   d  4  8  sas
```

- `dfT` DataFrame used `inplace=True` and it's index is shown as `Index(['a', 'b', 'c', 'd']`
  When `inplace=True`, index setting function will work on the original DataFrame and will return a modified DataFrame with changes.
- `dfF` DataFrame used `inplace=False` (Default) and it's index is shown as `RangeIndex(start=0, stop=4, step=1)`
  When `inplace=False`, index setting function will work on the original DataFrame but it will return the original DataFrame without changes.